اصول پردازش تصویر

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر مصطفی کمالی تبریزی

دانشکده علوم ریاضی

تمرین سری اول ــ سوال سوم

نام و نام‌خانوادگی: سیدعلیرضا خادم

شماره دانشجویی: ۹۷۱۰۰۳۹۸

موارد لازم.

برای اینکه کد $q_3.py$ به درستی اجرا شود، لازم است که تصویرِ ورودی با در مسیرِ EX1_Q3/images/ قرار بگیرد.بعد از اجرایِ کد از شما خواسته می‌شود که نام تصویر را همراه با فرمت آن وارد کنید، با وارد کردنِ نام تصویر ورودی و زدن کلید اینتر منتظر باشید تا کد اجرا شود و نتیجه در مسیرِ EX1_Q3/results/ ذخیره شود.

روند کلی حل.

ایده اصلی و روند کلی حل این سوال به این صورت است که، بعد از این که تصویر رقمی شده‌یِ ورودی را به سه قسمتِ افقیِ مساوی تقسیم کردم، برای هر یک از سه تصویر حاصل edge detection انجام میدهیم تا مرزها در این سه تصویر مشخص شوند. با توجه با اینکه این عکس ها از یک منظره هستند مرزهای مشترک آنها بسیار زیاد است و اگر بتوانیم این مرزهای مشترک را بر هم منطبق کنیم توانسته‌ایم این سه عکس رو به درستی رو هم قرار دهیم و نتیجه مطلوب را به دست آوریم. برای این که این کار را انجام دهیم یکی از این سه عکس را به عنوان مرجع در نظر میگیریم و هر سه عکس را روی این عکس با روش sum of squared differences سرچ می‌زنیم. و با توجه به نتایج میزان جابه‌جایی ها رو محاسبه می‌کنیم. ما در این سوال کانال آبی را به عنوان مرجع در نظر می‌گیریم و سه کانال را روی این کانال سرچ میزنیم. با فرض ثابت بودن کانال آبی جابه‌جایی کانال‌های سبز و قرمز به صورت زیر خواهد بود.

$$\Delta X_G = 9, \Delta Y_G = 88$$

$$\Delta X_R = 12, \Delta Y_R = 181$$

توضیح کد.

○ utilities.py

split_image_horizontally

این تابع یک عکس و عدد t را به عنوان ورودی می‌گیرد و عکس را افقی، به t قسمت مساوی تقسیم می‌کند و قسمت‌ها را در یک آرایه می‌ریزد و به عنوان خروجی این آرایه را برمی‌گرداند.

edge_detection_filter

این تابع یک عکس به عنوان ورودی می‌گیرد و فیلتر edge detection را روی این عکس اعمال می‌کند و نتیجه را با یک threshold باینری کرده و برمی‌گرداند.

ssd

این تابع یک عکس ، یک تمپلیت و عدد k را به عنوان ورودی می‌گیرد و با توجه به روابط زیر SSD را محاسبه می‌کند.

3. Sum of Squared Differences (SSD)

Can SSD be implemented with linear filters?

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

$$= \sum_{k,l} g[k,l]^2 + \sum_{k,l} f[m+k, n+l]^2 - 2 \sum_{k,l} g[k,l] f[m+k, n+l]$$

$$= C + (eyes * f^2) - 2\,(g * f)$$

با این تفاوت که ثابت C را محاسبه نمی‌کنیم و به تبع آن همه مقادیر را منهایِ مینیمم میکنیم تا مقدار منفی نداشته باشیم و به اضافه ۱.۱ می‌کنیم تا حداقل مقدار در result ۱.۱ باشد و بعد که میخواهیم لگاریتم بگیریم اعداد منفی نداشته باشیم و به مشکل نخوریم. بعد هم چون اعداد result خیلی بزرگ می‌شوند k بار لگاریتم می‌گیریم تا بتوانیم با این اعداد راحت تر کار کنیم. در نهایت هم مینیمم مقدار result را به عنوان نقطه‌ای که متچینگ انجام شده برمی‌گردانیم.

get_image_name

این تابع با چاپ کردنِ عبارتِ "please enter name of image with its format" از کاربر میخواهد تا نام عکسی که به عنوان ورودی قرار است به برنامه داده شود را همراه با فرمت تایپ کند. و بعد نامی که کاربر وارد کرده است را برمی‌گرداند.

q۳.py ○

در این فایل ابتدا نام تصویر از کاربر گرفته می‌شود و تصویر لود می‌شود. بعد تصویر به سه قسمت مساوی تقسیم می‌شود و روی هر کدام از این قسمت‌ها edge detection انجام می‌دهیم. edf_channel و eds_channel و edt_channel را با روش sum of squared differeces روی edf_channel سرچ می‌زنیم. بعد با نتایجی که به دست می‌آوریم و با ثابت فرض کردنِ کانالِ آبی میزان جابه‌جایی کانال‌هایِ سبز و قرمز را مجاسبه می‌کنیم و با استفاده از Geometric Transformations آن ها را جابه‌جا می‌کنیم. نیتجه از مرج کردنِ سه کانال حاصل می‌شود. در نهایت هم با توجه به اینکه حاشیه‌ها pixel هایی هستند که مقدار intensity یِ چنلِ آبی یا سبز یا قرمز در آنها نزدیک به ۲۵۵ است، حاشیه‌ها را حذف میکنیم و با تغییر عکس به unit8 آن را با فرمتِ jpg. در مسیرِ /EX1_Q1/results ذخیره می‌کنیم.